LISTING OF CLAIMS

1. (Previously presented) In a computer system, a method of generating a dispatch interface implementation, the method comprising:

receiving definition information that defines dispatch interface features of a dispatch interface, the dispatch interface including plural dispatch methods and one or more other methods;

receiving programming language code for the one or more other methods, each of the one or more other methods having a name;

generating a dispatch interface implementation for operating the one or more other methods, wherein the generating includes parsing the programming language code and the definition information during compilation, the dispatch interface implementation including:

executable code for the one or more other methods;

executable code for a first dispatch method, the first dispatch method for mapping the names to corresponding dispatch identifiers for binding at run time; and

executable code for a second dispatch method, the second dispatch method for calling the one or more other methods at run time responsive to client requests, each client request including a dispatch identifier.

2. (Original) The method of claim 1 wherein the definition information is embedded in a file for the programming language code.

3. (Original) The method of claim 1 wherein a file includes the programming language code and a statement for importing the definition information.

4. (Previously presented) The method of claim 1 wherein the generating comprises:

in the second dispatch method executable code, creating code for handling the arguments of one of the one or more other methods with a generic data structure.

5. (Original) The method of claim 1 wherein the dispatch interface implementation is part of a dual interface implementation, the method further comprising:

generating executable code for directly invoking the one or more other methods through a vtable mechanism at run time.

6. (Original) The method of claim 1 wherein the dispatch interface implementation further includes:

executable code for a third dispatch method, the third dispatch method for determining the availability of type information for the dispatch interface; and

executable code for a fourth dispatch method, the fourth dispatch method for retrieving available type information for the dispatch interface.

7. (Previously presented) A computer readable medium having stored thereon a computer executable compiler system that generates a late binding interface implementation from definition information and programming language code, the compiler system comprising:

a front end module that receives definition information and programming language code, the definition information defining late binding interface features of a late binding interface, the programming language code for implementing one or more late bound methods;

a converter module that identifies relations between the definition information and the one or more late bound methods during compilation, including parsing the definition information and the programming language code; and

a back end module that generates a late binding interface implementation based upon the relations, the late binding interface implementation for operating the one or more late bound methods.

8. (Original) The compiler system of claim 7 wherein the converter module identifies one or more relations, each relation between one of the one or more late bound methods and a corresponding identifier, and wherein based upon the one or more relations the back end module generates for each of the one or more late bound methods code mapping the name of the late bound method to the corresponding identifier for the late bound method.

9. (Original) The compiler system of claim 7 wherein the converter module identifies one or more relations, each relation between one of the one or more late bound

methods and a corresponding identifier, and wherein based upon the one or more relations the back end module generates for each of the one or more late bound methods code for calling the late bound method upon receipt of the corresponding identifier for the late bound method.

10. (Original) The compiler system of claim 7 wherein the converter module identifies one or more relations, each relation between type information and an argument of one of the one or more late bound methods, and wherein based upon the one or more relations the back end module generates code for handling the arguments of the late bound method with a generic data structure.

11. (Original) The compiler system of claim 7 wherein the converter module identifies a relation between a property indicator and one of the one or more late bound methods, and wherein based upon the relation the back end module generates code for retrieving or setting a corresponding property through the late bound method.

12. (Original) The compiler system of claim 7 wherein the late binding interface implementation is part of a combined early binding and late binding interface implementation, and wherein the back end module further generates an early binding interface implementation for the one or more late bound methods.

13. (Previously presented) A computer readable medium having stored thereon computer executable instructions for performing a method of automatically generating a late binding interface implementation, the method comprising:

receiving programming language code for one or more late bound methods of a late binding interface;

receiving definition information that defines late binding interface features of the late binding interface; and

generating a late binding interface implementation for operating the one or more late bound methods, wherein the generating includes parsing the programming language code and the definition information during compilation, the late binding interface implementation including

one or more late binding methods, wherein the one or more late binding methods include a first late binding method for calling the one or more late bound methods responsive to client requests.

14. (Original) The computer readable medium of claim 13 wherein the first late binding method lacks a call to a separate late binding interface implementation.

15. (Previously presented) The computer readable medium of claim 13 wherein the one or more late binding methods further include a second late binding method for mapping names of the one or more late bound methods to corresponding identifiers for run time binding, and wherein the second late binding method lacks a call to a separate late binding interface implementation.

16. (Previously presented) The computer readable medium of claim 13 wherein the one or more late binding methods further include a second late binding method for determining the availability of type information, and wherein the one or more late binding methods further include a third late binding method for retrieving available type information.

17. (Original) The computer readable medium of claim 13 wherein the definition information is embedded in a file for the programming language code.

18. (Previously presented) The computer readable medium of claim 13 wherein the generating comprises:

identifying type information for an argument of a first late bound method of the one or more late bound methods; and

for the implementation for the first late binding method, generating code for handling the argument with a generic data structure.

19. (Original) The computer readable medium of claim 13 wherein the late binding interface implementation adjoins an early binding interface implementation, the method further comprising:

generating the early binding interface implementation for directly invoking the one or more late bound methods.

20. (Previously presented) In a computer system, a method of automatically generating an interface implementation, the interface implementation having early binding and late binding mechanisms, the method comprising:

receiving programming language code for one or more methods of an interface;

receiving definition information that defines late binding interface features of the interface; and

generating an interface implementation for alternatively operating the one or more methods by an early binding mechanism or by a late binding mechanism, wherein the generating includes parsing the programming language code and the definition information during compilation, wherein the early binding mechanism provides for direct invocation of the one or more methods, and wherein the late binding mechanism provides for invocation of the one or more methods responsive to a request through a late binding method.

21. (Original) The method of claim 20 wherein the late binding mechanism maps names of the one or more methods to corresponding identifiers at run time.

22. (Previously presented) The method of claim 20 wherein the generating comprises:

identifying type information for an argument of a first method of the one or more methods; and

for the late binding mechanism, creating code for handling the argument with a generic data structure.

23. (Previously presented) In a computer system, a method of automatically generating client side call site code for calling a late bound method through a late binding interface, the method comprising:

receiving definition information for late binding interface features of a late binding interface;

receiving programming language code for calling a late bound method of the late binding interface;

parsing the definition information and the programming language code during compilation;

based upon type information for one or more input arguments of the late bound method, generating code for packing the one or more input arguments into a generic argument data structure; and

generating code for calling the late bound method through an invocation method of the late binding interface, wherein the calling includes passing the generic argument data structure to the invocation method.

24. (Original) The method of claim 23 further comprising:

based upon type information for a return value of the late bound method, generating code for unpacking the return value from a generic return value data structure.

25. (Original) The method of claim 23 further comprising:

generating code for calling a mapping method of the late binding interface, the mapping method associating a late bound method name with an identifier.

26. (Previously presented) The method of claim 1 wherein the second dispatch method has different arguments from each of the one or more other methods.

27. (Previously presented) The compiler system of claim 7 wherein the late binding interface implementation includes one or more late binding methods each having different arguments from the one or more late bound methods.

28. (Previously presented) The computer readable medium of claim 13 wherein the first late binding method has different arguments from each of the one or more late bound methods.

29. (Previously presented) The method of claim 20 wherein the late binding method has different arguments from each of the one or more methods.

30. (Previously presented) The method of claim 23 wherein the invocation method has different arguments from the late bound method.